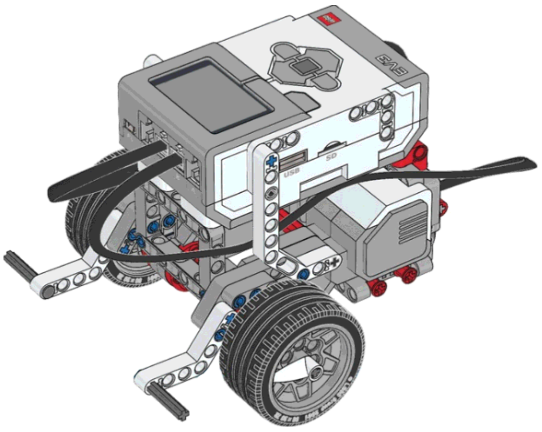
Lunar Rover

Software Project Management Plan

Software Engineering & Project



Team: PG-29
Benjamin Winding
Kin Leong Lee
Pavitterjeet Sidhu
Phan Huy Nguyen
Sean Hennessy
Xiaoshan Chen

05/09/2017

Contents

List of Figures

List of Tables

Revision History

Name	**Date**	**Version**	**Summary of Changes**
Ben	9-Sep-2017	0.10	Initial Draft
Huy Nguyen	10-Sep-2017	0.11	SMPM Section 7 Draft
Huy Nguyen	23-Sep-2017	0.12	Section 7 revise - add work breakdown diagram
Ben	2-Oct-2017	0.13	Revised the figures position
Pavi	20-Oct-2017	0.14	Changes in work breakdown structure, process model and corrected the Gantt Chart about time slot issue
Issac	27-Oct-2017	0.15	Add the description to describe section 7.3
Issac	27-Oct-2017	1.0	Release Final version

1 Introduction

1.1 Purpose and Scope

The purpose of this project involves building a prototype rover capable of surveying a designated area automatically, and developing an appropriately safe system for the client. The map shall be constructed in real time as long as the robot is surveying, showing lines, obstacles and current location of the robot. The final product relies on the existing EV3 LEGO Mind-storms robot provided by the client. It is to be controlled via a remote location, but is required to automatically make decisions based on the environment around it. The final product shall meet the requirements of client, which is included in the software specification document.

1.2 Assumptions and Constraints

Assumptions include:

- The group will contain 6 members throughout the project
- Everyone will complete the assigned sections
- Each member in the group are expected to spending 120 hours completing the project
- The functionalities of robot will be good during the client demonstrations

Constraints include:

- The system is to be implemented using the EV3 LEGO Mind-storms kit supplied by the client.
- The embedded software on the rover is to be written in Java, using the LeJOS Ev3 library version:0.9.0
- The external code used in the system shall not exceed 10%
- The build tool for compiling the software and deploying it to the system is ant.
- The system will be implemented using the wireless technology WiFi over an encrypted WPA2 connection.

1.3 Project Deliverables

The deliverables and due date of this project are list in the figure 1.

1.4 Evolution of the plan

The changes to the management plan will happen when:

- the definition of task is changed
- the group is not able to finish the task before deadline
- the group complete the task ahead of deadline

The project manager will evaluate the performance when task is completed and will held a group meeting with group members to review issues. When the task is completed ahead of deadline, tasks whose deadline is in the future can be added in advance.

Deliverables	Due Date
Software Requirement Specification 1st Draft	22 Aug, 2017
Software Project Management Plan 1st Draft	5 Sep, 2017
Milestone 1 demo specification finalize and sign agreement	5 Sep, 2017
Milestone 1 demo	12 Sep, 2017
Milestone 2 demo specification finalize and sign agreement	12 Sep, 2017
Software Design Document Draft	3 Oct, 2017
Milestone 2 demo	3 Oct, 2017
User manual and instruction draft	17 Oct, 2017
Release final version of SRS, SPMP, SDD and user manual	24 Oct, 2017
Code free and product release	24 Oct, 2017
Final presentation	31 Oct, 2017

Figure 1: The due date of project deliverables

2 References

- 'A Guide to the Project Management Body of Knowledge' 2013, 5th edn, Project Management Institute
- IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998
- Google Java Style Guide, 2016, https://google.github.io/styleguide/javaguide.html

3 Definition

- **SRS-**Software Requirements Specification
- **SPMP-**Software Project Management Plan
- **SDD-**Software Design Document
- **GUI-**Graphic User Interface
- **IDE-**Integrated Development Environment

4 Project Organisation

4.1 The role and responsibilities

The Software project team consists of Project manager, Documentation Manager, Testing Manager, IT Manager, Development Manager and Quality Manager. The team member's role and responsibilities are listed as below.

- **Name: Pavitterjeet Sidhu**
- Role: Project Manager
- Responsibility:
  1. Planning all activities throughout the project life cycle, assigning each team member's duties and monitoring their progress.
  2. To organise internal and client meetings.

3. To ensure the project moving along on the right path and meet the deadline.
- Reason: Pavi has experience in leading a project. He is familiar with the project's process model.

- **Name: Benjamin Winding**
- Role: Documentation Manager
- Responsibility:
  1. To integrate team member's documentation work and design project structure.
  2. To review team member's document such as SPMP, SRS and SDD and ensure its the quality with acceptance level.
- Reason: Ben has experience of using latex and github. He can identify the issue of our latex file quickly and be familiar with using github to make a file structure.

- **Name: Kin Leong Lee**
- Role: Testing Manager
- Responsibility:
  1. To design a set of testing cases and testing scripts for Graphic User Interface and Robot's functionalities.
  2. To review all testing results and write a testing report.
- Reason: Kin has experience of working in IT company and performed difference testing for software product. He is familiar with the testing process and building testing script.

- **Name: Xiaoshan Chen**
- Role: IT Manager
- Responsibility:
  1. To be responsible for the connection between EV3 robot and remote system.
  2. To handle and solve all the issues about using IDE.
- Reason: Shan has experience of working a IT company and responsible for IT support. She can handle difference IT problems and has experience of using intellij.

- **Name: Phan Huy Nguyen**
- Role: Development Manager
- Responsibility:
  1. To design the program structure and develop coding policies.
  2. To review all team member's coding work including GUI, robot movement, robot sensors and path finding.
- Reason: Phan has software company working experience. He is familiar with the industry standard of coding.

- **Name: Sean Hennessy**
- Role: Quality Manager
- Responsibility:
  1. To identify all the risks of project throughout project life cycle.
  2. To plan the strategic to solve the project issues and ensue all the submitted works following the standard and good quality.

- Reason: Sean has experience of making budget plan for the project. He also studied project management course and has a skill to identify the risks of project.

The detail responsibilities for each team member is shown on Table1:

Table 1: Responsibilities table

Tasks/Members	Pavi	Ben	Issac	Shan	Phan	Sean
Management						
Planning and Schedule	A&R	I	I	I	I	I
Risk Management	I	I	I	I	I	A&R
Meeting Agenda	R&A	R	R	R	R	R
Meeting minutes	R&A	R	R	R	R	R
Documentation						
Documents (SRS, SPMP, SDD)	R	R&A	R	R	R	R
Review Documents	I	R&A	I	I	I	I
Connection Management						
Establish connection between robot and remote system	I	I	R	R&A	I	I
Implement plugin to IDE and make it work	I	I	I	R&A	I	R
Software Development						
Developing GUI	R	I	R	I	R&A	R
Developing Robot functionalities	I	R	I	R	R&A	I
Integration	R	R	I	I	R&A	I
Testing						
Building Testing script	I	R	R&A	I	R	I
Acceptance Test	R	I	R&A	R	I	R

- R(Responsible): Team member is responsible for this task
- A(Accountable): Team member is responsible to assign the work to each group member
- I(Informed): Team member will be informed when the task is completed (A Guide to the Project Management Body of Knowledge 2013 p.262)

5 Risk Management

5.1 Introduction

This risk management plan is implemented to identify and apply mitigation strategies to potential risks of the project. The types of risks that will be identified will be those that may affect the health or safety of the personnel working on the project as well as any risks to the expected delivery of the project. These risks will be categorised and given an impact and probability rating.

5.2 Identification

Types

There will be 5 risk categories identified. These are categories are follows:

- Technology
- People
- Tools
- Requirements
- Estimation

Priority convention

There will be 3 main probability and impact descriptors

- Low- probability:10-30% , impact: 0-1 days project delay
- Medium - probabilty: 30-60%, impact: 2-4 days project delay
- High- probability: 60-100%, impact: > 4 days project delay

5.3 Analysis and Planning

Technology Risks

Robot or parts damaged or lost
Probability: Low
Impact: High
Indicators: Unsecure sotrage location.
Strategy: If any parts or the whole robot is damaged or lost, the importance of the lost part is assessed by the team as a group. If necessary, the team must seek to collect funds in order to buy replacement parts.

Wifi cannot handle code being sent to robot
Probability: Low
Impact: High
Indicators: Increase in communication delay
Strategy: Every week robot must be tested so that it can comply to commands via a wired cconnection

People Risks

Team members get sick at critical times
Probability: High
Impact: Medium
Indicators: Team members in poor health or mood
Strategy: Ensure any sick team members are encouraged to stay home in order to avoid spreading to other members. Project manager is to ensure that task assignment has some overlap in order to ensure there is at least one person able to work on critical sections.

Team members leave the course
Probability: Low

Impact: High
Indicators: self evident
Strategy: Project manager to ensure that there is overlap in task assignment and that all team members have some expereience in every aspect of code. Ask team members to give as much warning as possible.

Tools

Final product unable to run on certain OS or environment
Probability: Low
Impact: High
Indicators:Testing shows intermittent issues on non-standard OS or no testing done on different OS.
Strategy: Team must ensure that they bring a laptop which is tested to work with the code for final presentation in case the client's hardware is unable to run the software.

University Github server no longer supported
Probability: Low
Impact: High
Indicators: public announcement or github GUI/Command line no longer working.
Strategy: Each team member to ensure that the project is pulled at least once per day on their machine. In the event of Github no longer being available, alternate version control methods such as SVN may be considered.

Team members update to different versions of IDE or JDK
Probability: Low
Impact: Low
Indicators:Compile errors with certain members. Errors after certain members update the code.
Strategy: Team members are to be informed that unless specified by the project manager, the IDE and JDK are not to be updated and must remain the same accross the length of the project.

Requirements

Client dictates major changes to requirements
Probability: Low
Impact: High
Indicators:Multiple consecutive change requests. Client complaints.
Strategy: communications to the client in weekly meetings are to be clear in what the team is working on and how it fits into the current requirements so that any changes the client would like to make can be brought up incrementally. Project manager is to ensure that client is aware that major requirements changes will require significant increase in budget to compensate.

Estimation

Time required to develop is underestimated
Probability: Low
Impact: Low to High
Indicators:failure to meet internal milestones.
Strategy: Team members are to inform the rest of the group if they feel that the assigned task is actually larger than anticipated. Project manager is to ensure that they check on the progress of the team regularly and is not soley reliant on team feedback.
Debugging and testing undersestimated
Probability: Medium
Impact: Low
Indicators:tests unable to show faults. unable to meet testing milestones.
Strategy: Testing manager is to inform the group if they feel that testing will fall behind so they may gather additional support from the rest of team. Individual members must ask for help if they feel that they may lag behind while coding their assigned task.

5.4 Monitoring

Additional risks may be added to risk management plan by the risk assessment manager. Documentation of risks should be added to the SPMP document revision history. Current risks may be reassessed in terms of severity or probabilty and this shall be undertaken by the risk manager fortnightly.

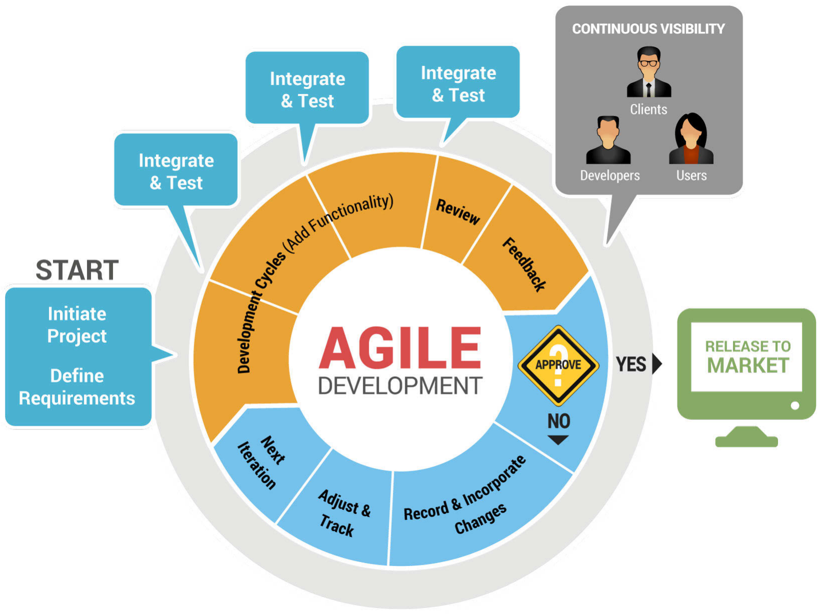


Figure 2: Agile process model diagram

6 Process Model

In this project, we are following agile process development model as shown in figure 2 because this project requires adaptive planning along with evolutionary development and continuous improvement. The initial phase of the project is focused on gathering the software requirement.There will be two official milestones and several internal project team milestones, in order to deliver the product on time. Project deliverables are divided into sub-activities and for each activity a development cycle will be followed which will include different phases such as design, development, testing, review and improvement.

Agile process development model (Shown on Fig.2) provides high customer satisfaction due to continuous delivery of the software modules and ensures the high quality software development. This model has advantages such as no planning is required to start the project as compared to other traditional software development models such as water fall in which project team makes strict promises to the client by doing requirement analysis and planning. This model is very easy to manage and provides great flexibility to the project team and client.

For low level development and management, we used Scrum framework which is also a type of agile model. We decided to use this model because in this every one wants to get some experience in different phase of software development such as design, development , testing etc.

7 Work Plan

7.1 Work Activities

7.1.1 Software Requirement Specification

- SRS-T001: SRS Draft
  - SRS-T001-1: Brainstorming meeting for ideas.
  - SRS-T001-2: Understand and develop user requirement.
  - SRS-T001-3: Define product features and functional requirement.
  - SRS-T001-4: Develop external interface and non-functional requirements.
  - SRS-T001-5: SRS draft finalization meeting.
- SRS-T002: SRS Final Version
  - SRS-T002-1: Revise SRS draft following client's recommendations after SRS draft presentation meeting.
  - SRS-T002-2: Keep track client requirements change every weeks.
  - SRS-T002-3: Team review meeting before release official SRS document.

7.1.2 Software Product Management Plan

- SPMP-T001: SPMP Draft
  - SPMP-T001-1: Brainstorming meeting for ideas.
  - SPMP-T001-2: Develop project organization, specific role and responsibility for each member in this project.
  - SPMP-T001-3: Develop risk management plan.
  - SPMP-T001-4: Analysis team characteristics and select appropriate process model.
  - SPMP-T001-5: Create working plan based on current SRS, process model and team capacity.
  - SPMP-T001-6: SPMP draft finalization meeting.
- SPMP-T002: SPMP Final Version
  - SPMP-T002-1: Revise SPMP draft following client's recommendations after SPMP draft presentation meeting.
  - SPMP-T002-2: Weekly update if there is with any change in process model, resources, development plan etc.
  - SPMP-T002-3: Team review meeting before release official SPMP document.

7.1.3 Software Design Document

- SDD-T001: System Design
  - SDD-T001-1: Develop system overview of product.
  - SDD-T001-2: Define system components base on system overview.
  - SDD-T001-3: System components design details for each component
- SDD-T002: GUI Design
  - SDD-T002-1: Analyst user requirements and system features.
  - SDD-T002-2: Define GUI appropriate with each features.

  - SDD-T002-3: Create the mock-GUI and test it feasibility and user experience.
- SDD-T003: SDD Final Version
  - SDD-T003-1: Revise SDD draft following client's recommendations after SDD draft presentation meeting.
  - SDD-T003-2: Weekly update if there is with any change in requirements, features which need to be reflect into system design, GUI design etc.
  - SDD-T003-3: Team review meeting before release official SDD document.

7.1.4 Implementation

- IMP-T001: Prototyping
  - IMP-T001-1: Create simple manual control with UI.
  - IMP-T001-2: Create autonomous controller make robot move randomly with human control.
  - IMP-T001-3: Enhance from IM-T001-2 make robot move follow a track to test color sensor.
  - IMP-T001-4: Enhance from IM-T001-2 make robot automatically moving while detect and avoid obstacle.
- IMP-T002: GUI Implementation
  - IMP-T002-1: Create GUI from mock-up design of SDD-T002.
  - IMP-T002-2: Integration prototyping with new GUI.
  - IMP-T002-3: Implement user input feature on GUI. Allow user to manually add new information such as NGZ, destination, landing site etc.
- IMP-T003: Implement autonomous mode.
  - IMP-T003-1: Implement path finding algorithm.
  - IMP-T003-2: Enhance IMP-T001-3 and IMP-T001-4 so that robot can recognize and follow the track.
  - IMP-T003-3: Enhance IMP-T001-3 and IMP-T001-4 so that robot can recognize physical obstacle, NGZ, cratters etc.
  - IMP-T003-3: Implement area map survey strategy by moving zig-zag while detect and avoid obstacles, NGZ, cratters etc.
- IMP-T004: Implement manual control mode.
  - IMP-T004-1: Implement warning and confirmation when user control robot into danger region.
  - IMP-T004-2: Implement emergency stop feature.
  - IMP-T004-3: Implement autonomous/manual switch.
- IMP-T005: Map Display and Construction.
  - IMP-T005-1: Display import map xml info onto GUI-MAP.
  - IMP-T005-2: Display new informations return from sensor on GUI-MAP.
  - IMP-T005-3: Display warning on GUI-MAP when robot move into danger region.
  - IMP-T005-4: Implement export result survey xml map.

- IMP-T006: Team final review.
  - IMP-T006-1: Each member review all implemented features.
  - IMP-T006-2: Final review meeting and collect team features feedback.
  - IMP-T006-3: Make change base on feedback.

7.1.5 Testing

- Test-T001: Unit Test.
  - Test-T001-1: Create unit test for all major computation component.
  - Test-T001-2: Regular review and make test case up to date.
- Test-T002: Integration Test.
  - Test-T002-1: Define test cases and test-acceptance criteria base on requirement and scope define in SRS.
  - Test-T002-2: Meeting with client and finalize all test cases.
  - Test-T003-3: Perform test product functionalities base on defined test cases while implement process.
  - Test-T003-4: Keep track all change in requirement and update corresponding test cases.
  - Test-T003-5: Recording all fail tests and keep track along side with development process.

7.1.6 User Manual

- UM-T001: Define user manual structure.
  - UM-T001-1: Team brainstorming for user manual ideas.
  - UM-T001-2: Create user manual draft.
  - UM-T001-3: Present and get feedback from clients.
- UM-T002: Finalize user manual.
  - UM-T001-1: Review client feedback and revise user manual.
  - UM-T002-2: Team meeting and release final version.

7.1.7 Release

- RL-T001: Pre-Relase
  - RL-T001-1: Run over all test case define in Test-T001 and Test-T002.
  - RL-T001-2: Review and check is there any existing bug.
  - RL-T001-3: Fix all fail test cases and existing bug.
- RL-T002: Create review checklist.
- RL-T003: Go over release checklist and deploy final product to client.

7.2 Milestone

7.2.1 Internal Milestone

- IMS-001: Robot manual control with basic movement.
  - Due Date: End of week 3.
  - Description:
    - Simple GUI with buttons : Up, Down, Left, Right, Stop.
    - Robot move as expected when use control button on GUI.

- IMS-002: Software Requirement Specification Draft.
  - Due Date: End of week 4.
  - Description:
    - Have first draft of SRS document follow SRS template with all sections in detail.
    - Document must be in LaTEX file.
    - Document manager must perform proof-reading over whole document and edit spelling, grammar, formating mistake if any.
- IMS-003: Software Project Management Plan Draft
  - Due Date: End of week 6.
  - Description:
    - Have first draft of SPMP document follow SPMP template with all sections in detail.
    - Document must be in LaTEX file.
    - Document manager must perform proof-reading over whole document and edit spelling, grammar, formating mistake if any.
- IMS-004: Mock-up of final GUI
  - Due Date: End of week 6.
  - Description:
    - Finish GUI design with all GUI component which user need to finish their mission.
    - Go through SRS user requirement, system features make sure all requirement can perform easy on GUI.
    - Some buttons, features or component on GUI may not working since it just placeholder.
- IMS-005: Software Design Document Draft
  - Due Date: End of week 8.
  - Description:
    - Have first draft of SPMP document follow SPMP template with all sections in detail.
    - Document must be in LaTEX file.
    - Document manager must perform proof-reading over whole document and edit spelling, grammar, formating mistake if any.
- IMS-006: Robot autonomous survey mode
  - Due Date: End of week 9.
  - Description:
    - Robot can automatically move inside survey area.
    - Robot can detect and avoid obstacle, cratter, NGZ and avoid them.
    - Robot can collect sensor data and display it on GUI-Map.
    - Robot can move to point define by user.
    - Robot can move follow track.
- IMS-007: Finish final GUI functionalities
  - Due Date: End of week 10.
  - Description:

        * Every function on GUI work as define in SDD user interface section.

- IMS-008: Integration and Testing
    - Due Date: End of week 11.
    - Description:
        * Integrate manual control mode and autonomous mode in to GUI.
        * Verify all functionalities still working as expected and match with requirement in SRS document.
        * Switch between manual/autonomous without any problem.
        * Robot can complete survey in less than 20 minutes and return to landing site.
- IMS-009: Prepare for final presentation with client.
    - Due Date: End of week 12.
    - Description:
        * All remaining must be solve.
        * All test case must pass.
        * Code review already perform and have a clear code base with sufficient up to date comment.
        * Final code must be submit before code freeze deadline.
        * Slide or Postcard must ready if there is required them.

7.2.2 Client Official Milestone

- OMS-001: Client Official Milestone 1 Demo.
    - Due Date: End of week 7.
    - Description:
        * Finish GUI design with all GUI components which user need to finish their mission.
        * Robot movement display correct on Map-GUI.
        * Robot can move randomly and avoid obstacles and NGZ.
        * Some buttons, features or component on GUI may not working since it is just placeholder.
- OMS-002 : Client Official Milestone 2 Demo.
    - Due Date: End of semester break.
    - Description:
        * Robot can move survey the area with effective strategy such as move cycle or zig-zag.
        * User can manual define NGZ and robot can recognize NGZ, obstacle, crater to avoid them while survey.
        * New informations of survey area are display in real-time on GUI-Map.

7.3 Work Breakdown Structure

The Fig.3 shows a work breakdown structure of this project. This project consists of 5 difference phases: Verfication of requirement, Project management plan development, Software development, Testing and product release.

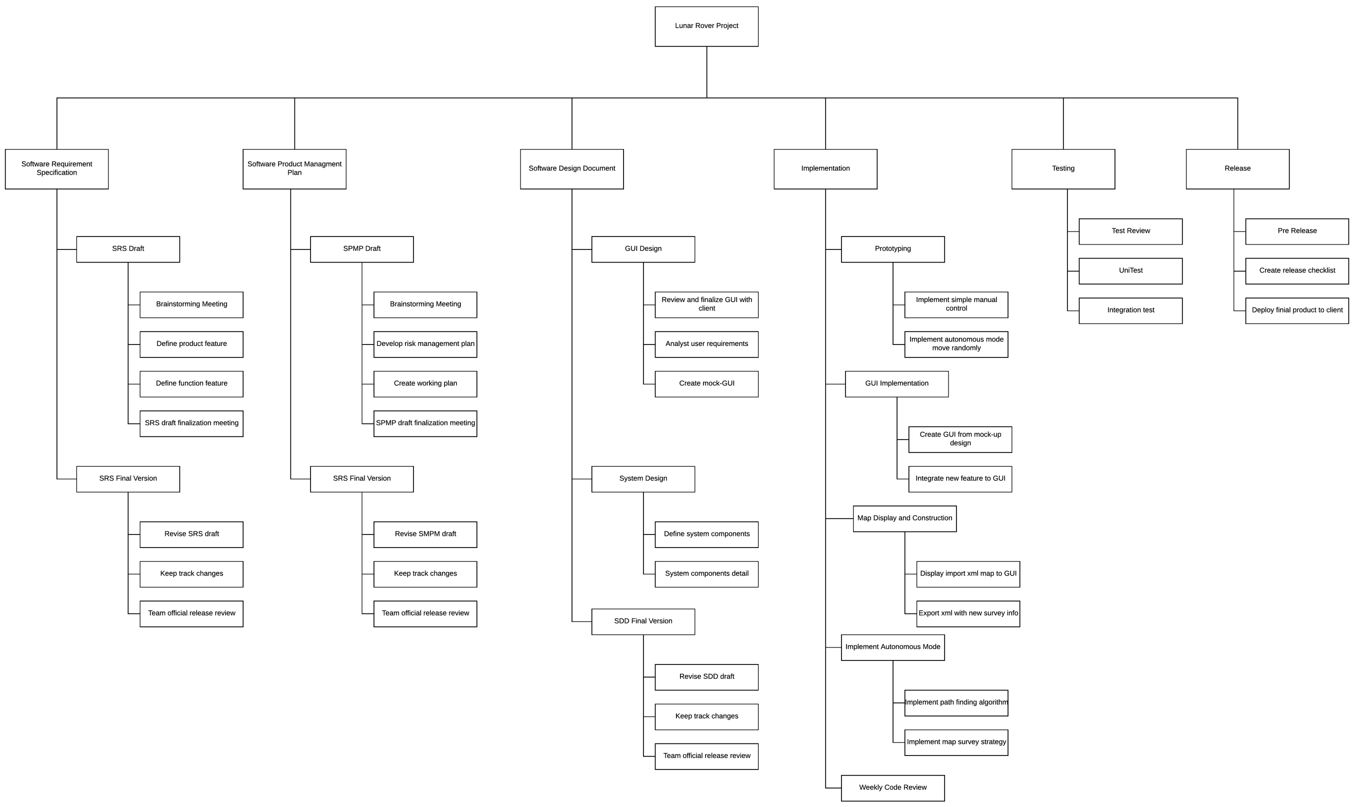

Lunar Rover Project
Software Requirement Specification
SRS Draft
Brainstorming Meeting
Define product feature
Define function feature
SRS draft finalization meeting
SRS Final Version
Revise SRS draft
Keep track changes
Team official release review
Software Product Managment Plan
SPMP Draft
Brainstorming Meeting
Develop risk management plan
Create working plan
SPMP draft finalization meeting
SRS Final Version
Revise SMPM draft
Keep track changes
Team official release review
Software Design Document
GUI Design
Review and finalize GUI with client
Analyst user requirements
Create mock-GUI
System Design
Define system components
System components detail
SDD Final Version
Revise SDD draft
Keep track changes
Team official release review
Implementation
Prototyping
Implement simple manual control
Implement autonomous mode move randomly
GUI Implementation
Create GUI from mock-up design
Integrate new feature to GUI
Map Display and Construction
Display import xml map to GUI
Export xml with new survey info
Implement Autonomous Mode
Implement path finding algorithm
Implement map survey strategy
Weekly Code Review
Testing
Test Review
UniTest
Integration test
Release
Pre Release
Create release checklist
Deploy finial product to client

Figure 3: Work Breakdown Structure

7.4 Schedule Allocation

7.4.1 Task schedule

The task scheduling will be shown in detail by the Gantt chart below. In summary, the Gantt chart will indicate which team member has responsibility for a task. The time allocation for each task with a specific deadline. Tasks can work in parallels are also visualize and can easy to find in the chart.

7.4.2 Major Task and Deadline Time

Deadline Time	**Task name**
Week 5	Software Requirement Specification Draft
Week 7	Software Project Management Plan Draft
Week 7	Milestone 1 demo specification finalize and sign agreement
Week 8	Milestone 1 demo
Week 8	Milestone 2 demo specification finalize and sign agreement
Week 9	Software Design Document Draft
Week 9	Milestone 2 demo
Week 11	User manual and instruction draft
Week 12	Release final version of SRS, SPMP, SDD and user manual
Week 12	Code free and product release

8 Supporting Plans

8.1 Configuration Management Plan

This software project consists of 5 official documents including SRS, SPMP, SDD, Testing report and User Manuel), meeting agenda and minutes, and software for operating EV3 robot. All these configuration items' standards are developed by documentation manager and monitored by quality manager. All team members are responsible for following this configuration standard during the whole project life cycle. The detail of software and testing components will be mentioned in SDD document and testing report.

8.1.1 GitHub Repository structure and Configuration control

The repository structure is built based on the component of the project. The highest level of repository is Code and Documentation.

- `...\2017-S2-SEP-PG29\Documentation`
- `...\2017-S2-SEP-PG29\Code`

The detail GitHub Repository structure is shown on fig.4.

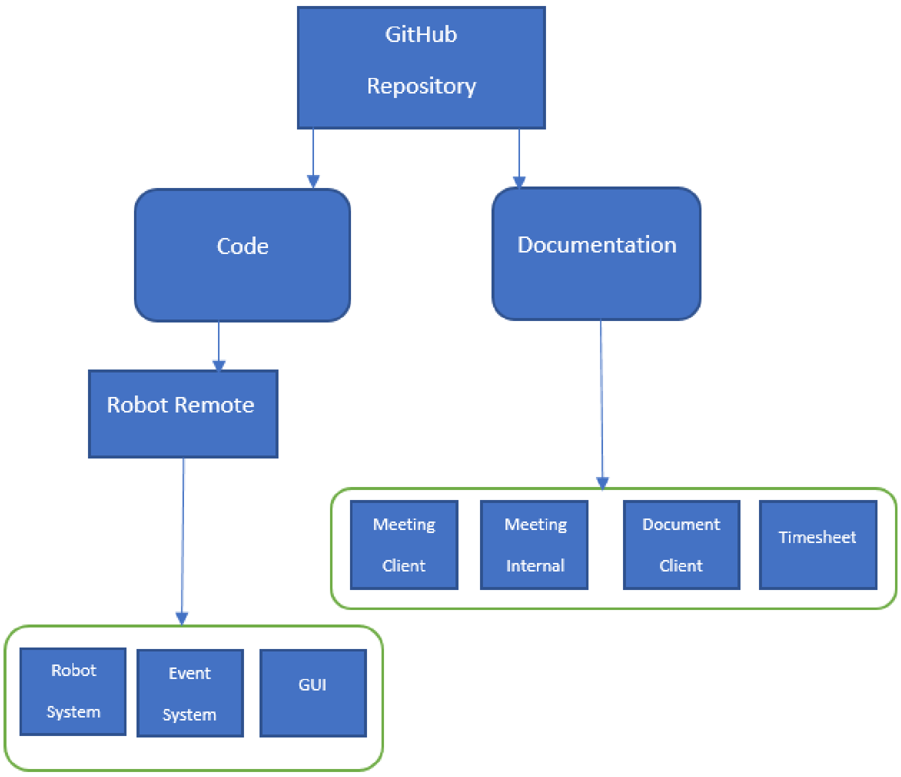


Figure 4: GITHUB repository

There are three configuration directories which have special control for team members to edit. The Software components are not included in this section and

Table 2: Revision history example

Name	Date	Version	Summary of Changes
1st Draft	21/8/2017	0.9.1	Initial Draft
2nd Draft	22/8/2017	0.9.2	Identification numbers added

will be discussed in detail on SDD document.
The lists of configuration control are shown as below:

Library control:
This library directory is for development purpose. There shall allow to store jar file only. No other file format is allowed. Team members are require to get the approval from development manager to add new jar file.

- `...\Code \UiRemote \lib`

Media control:
Those directory contain png file only. No other file format is allowed. Team members can feel free to add any png file for development or development purpose. Team members shall commit the change with proper message to indicate which files they are added.

- `...Code\UiRemote \src \RobotRemote \UI \Views`
- `...Documentation\Views`

Documentation control:
Those directories contain tex and pdf file only. Team members are not allowed to add any file in those directories. Only documentation manager can add new file to those directories

- `...Documentation\SRS`
- `...Documentation\SPMP`
- `...Documentation\SDD`

8.1.2 Configuration Naming System and version control

Any draft of configuration items shall be followed as below naming standards:

- `<Title>_<version>.<extension>`
  `For examples:  SRS_v0.9.tex`
- The Final version is v1.0. The draft version will be named as v0.1-v0.9.
- The modified date for each sub-version shall be mentioned in revision history table which is included into documents.

The examples of revision history for SRS-v0.9.tex is shown on Table.2 :

The revision table for final version shall include all the change for each version. After final version released, any change is required a change request which is mentioned in section 8.3.

The meeting agenda and meeting minutes shall follow as below naming system:
`<Title>_<Type>_<week>.<extension>`

- Title: Agenda or Minutes
- Type: Client or Internal

```
For example:  agenda_client_week3.tex, minutes_internal_week3.tex
```

Usually, there are only one client meeting and internal group meeting per week. If there are any emergency meeting, the date should be included.

```
For example:  agenda_internal_week3_17082017.tex
The date format shall be <days><months><year>
```

8.1.3 Change request handling

There are two types change request handling. Documentation and Software change request handling.

Documentation:
The Documents will be divided by documentation manager into difference subtasks for each team member to edit their own parts.

- Team members can make any change on their own part before the final version release.
- Team members is required to edit the revision table to record their change on the document.
- The documentation review meeting will be held by documentation manager to review all the changes on the document before releasing final version.
- The change on Final version's document is not allowed without discussion in another new review meeting.

Software:

- Team member must need to create new branch in github repository to make a change or add new function on software.
- The change on this branch must not affect master branch.
- Regular software review meeting will be held by development manager to review all the team member's work and discuss the potential issue before merging the work to master branch.

8.2 Documentation Plan

The documentation plan involves several key documents and revision processes that ensure the quality of the overall project is kept at a high level.

8.2.1 SRS

The Software Requirements Specification will be constructed using the *IEEE Recommended Practice for Software Requirements Specifications* standard provided by the IEEE.

Structure

- Revision History
1. Introduction
2. Overall Description
3. User Requirements
4. System Features
5. External Interface Requirements
6. Other Non-Functional Requirements
7. Other Requirements
- Appendix A: Glossary
- Appendix B: Analysis Models
- Appendix C: Issues List

8.2.2 SPMP

The Software Project Management Plan shall be constructed with the following structure:

Structure

- Revision History
1. Introduction
2. References
3. Definition
4. Project Organisation
5. Risk Management
6. Process Model
7. Work Plan
8. Supporting Plans
9. Appendices

8.2.3 SDD

The the Software Design Document is as follows:

Structure

- Revision History
1. Introduction
2. System Overview
3. System Architecture and Components Design
4. Data Design
5. Design Details

6. Human Interface Design
7. Resource Estimates
8. Definitions, Acronyms, and Abbreviations
9. Appendices

8.2.4 Client Meetings

The formal documentation for client meetings is crucial as it is a record of what ideas and decisions were made that will affect the entire project.

Agenda
An important document for client meetings is the client meeting agenda. This document provides both parties with a common understanding of what purpose of the meeting is and the items that will be discussed. Structure of the agenda is as follows:

- Attendance
- Progress summary of the project
- Points from last meeting
- New questions for this meeting
- Questions for the project team

Minutes
Prior to each client meeting, a group member will be designated to record the minutes of the meeting. They're task is to record as much important information in the meeting as they can including but not limited to the following:

- Attendance
- Questions we asked the client, and answers
- Questions the client asked us
- Problems needed to be addressed for next meeting
- Ideas, or possible solutions during discussion

8.2.5 Internal Meetings

Although internal meetings are less formal, we plan to produce an meeting agenda and record meeting minutes. Similar to the client meeting agenda, the internal meeting agenda format should allow for everyone to prepare for the meeting and provide a much more efficient discussion. Additionally, the internal versions will contain enough information that people can refer to at a later date.

8.2.6 Timesheets

Another internal document is the group member timesheet. This is a document which is used for internal tracking of group member participation and progress. It should be a record of the time spent on what tasks, and should be filled out on a weekly biases and saved in the Github repository.

8.2.7 Document Review Processes

Several revision processes will be used in order to enhance the quality of the documents planned out. These processes include:

- Brainstorming
- Spelling and Grammar Review
- Collaborative and traceable document editing through Github with latex
- Team member review

8.3 Quality Assurance Plan

The quality assurance of the project is managed using a combination of several methods. The following subsections detail the plans used throughout the project.

8.3.1 Coding Standards

Throughout the project a common set of coding conventions are to be used. This includes the following points:

Code Formatting
All code that is committed to the repository is to be formatted with respect to the *Google Java Style Guide* This is automated through the IDE chosen, which is Jetbrains IntelliJ 2017.

8.3.2 Software Review Process

Unit Testing
With an effort to provide quality code that has proven functionality, a testing plan will be carried out, in which all functional code will have associated unit tests.

Code Review
Finally, all code that is committed to the repository is to be reviewed before it is merged to the master branch.

8.3.3 Traceability

Github
To ensure traceability throughout the duration of the project, the Github repository will be the primary source of information. This will provide a detailed history of changes to each document and which team members made them.

Slack
Additionally, Slack will be used for internal correspondence between the project team. This also allows for traceability of conversations between the team, which helps document ideas and features for the project.

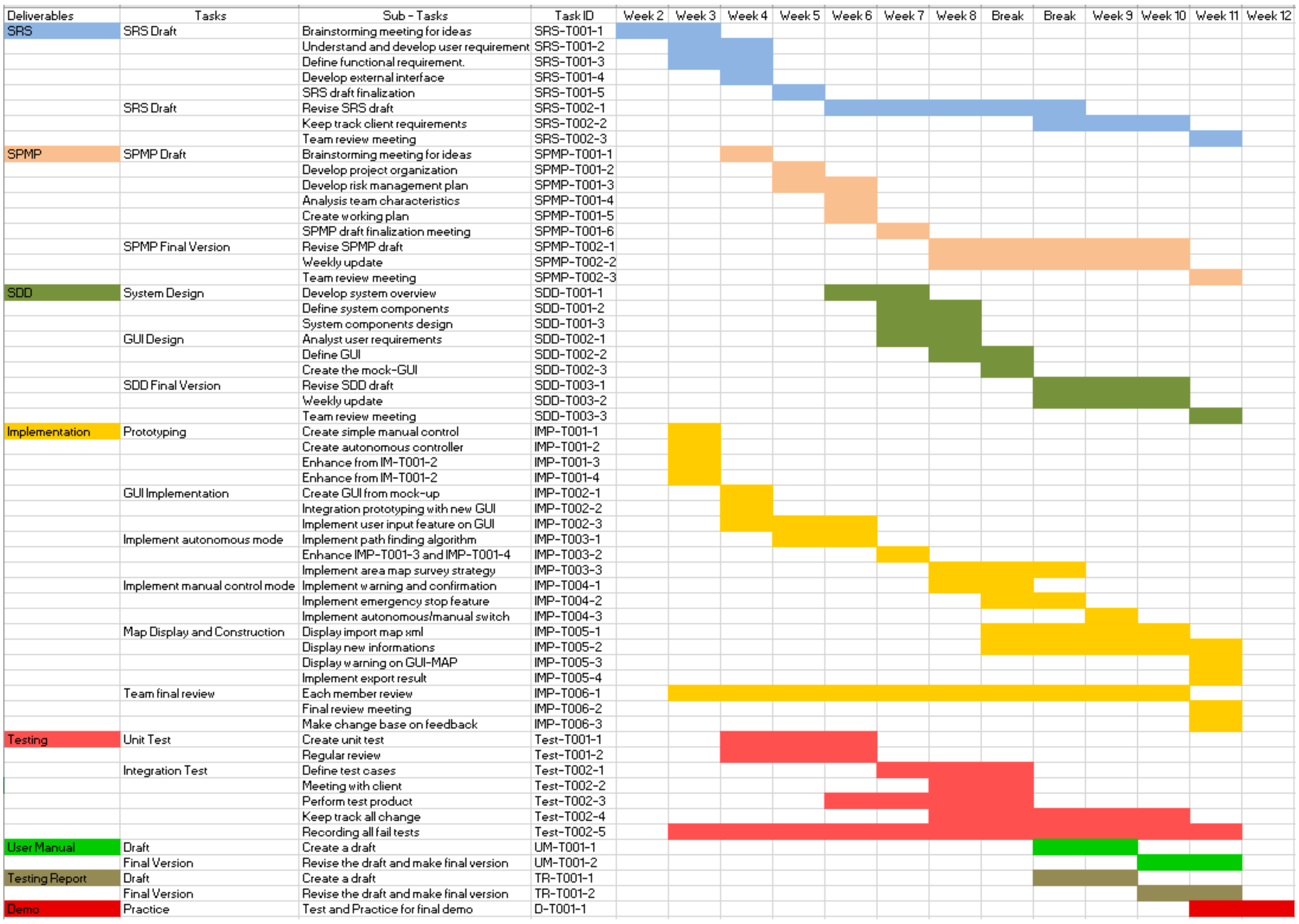
8.3.4 Roles

As described in section 4.1, each group member is responsible for a different area of the project. This dependency on specialised roles, is offset by the processes used to improve the quality of the project, as all group members will get the chance to work with the code, testing, documentation and project management. This additionally helps with redundancy, and improves the resilience of the project, with respect to unforeseen changes.

Pavi	- Project Manager
Ben	- Documentation Manager
Issac	- Testing Manager
Sammy	- IT Manager
John	- Development Manager
Sean	- Quality Manager

9 Appendices

A Gantt Chart - Overview



Deliverables	Tasks	Sub - Tasks	Task ID	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Break	Break	Week 9	Week 10	Week 11	Week 12
SRS	SRS Draft	Brainstorming meeting for ideas	SRS-T001-1	🟦	🟦											
		Understand and develop user requirement	SRS-T001-2		🟦	🟦										
		Define functional requirement.	SRS-T001-3		🟦	🟦										
		Develop external interface	SRS-T001-4			🟦										
		SRS draft finalization	SRS-T001-5				🟦									
	SRS Draft	Revise SRS draft	SRS-T002-1					🟦	🟦	🟦	🟦	🟦				
		Keep track client requirements	SRS-T002-2									🟦	🟦	🟦		
		Team review meeting	SRS-T002-3												🟦	
SPMP	SPMP Draft	Brainstorming meeting for ideas	SPMP-T001-1			🟧										
		Develop project organization	SPMP-T001-2				🟧									
		Develop risk management plan	SPMP-T001-3				🟧	🟧								
		Analysis team characteristics	SPMP-T001-4					🟧								
		Create working plan	SPMP-T001-5					🟧								
		SPMP draft finalization meeting	SPMP-T001-6						🟧							
	SPMP Final Version	Revise SPMP draft	SPMP-T002-1							🟧	🟧	🟧	🟧	🟧		
		Weekly update	SPMP-T002-2							🟧	🟧	🟧	🟧	🟧		
		Team review meeting	SPMP-T002-3												🟧	
SDD	System Design	Develop system overview	SDD-T001-1					🟩	🟩							
		Define system components	SDD-T001-2						🟩	🟩						
		System components design	SDD-T001-3						🟩	🟩						
	GUI Design	Analyst user requirements	SDD-T002-1						🟩	🟩						
		Define GUI	SDD-T002-2							🟩	🟩					
		Create the mock-GUI	SDD-T002-3								🟩					
	SDD Final Version	Revise SDD draft	SDD-T003-1									🟩	🟩	🟩		
		Weekly update	SDD-T003-2									🟩	🟩	🟩		
		Team review meeting	SDD-T003-3												🟩	
Implementation	Prototyping	Create simple manual control	IMP-T001-1		🟨											
		Create autonomous controller	IMP-T001-2		🟨											
		Enhance from IM-T001-2	IMP-T001-3		🟨											
		Enhance from IM-T001-2	IMP-T001-4		🟨											
	GUI Implementation	Create GUI from mock-up	IMP-T002-1			🟨										
		Integration prototyping with new GUI	IMP-T002-2			🟨										
		Implement user input feature on GUI	IMP-T002-3			🟨	🟨	🟨								
	Implement autonomous mode	Implement path finding algorithm	IMP-T003-1				🟨	🟨								
		Enhance IMP-T001-3 and IMP-T001-4	IMP-T003-2						🟨							
		Implement area map survey strategy	IMP-T003-3							🟨	🟨	🟨				
	Implement manual control mode	Implement warning and confirmation	IMP-T004-1							🟨	🟨					
		Implement emergency stop feature	IMP-T004-2								🟨	🟨				
		Implement autonomous/manual switch	IMP-T004-3										🟨			
	Map Display and Construction	Display import map xml	IMP-T005-1								🟨	🟨	🟨	🟨		
		Display new informations	IMP-T005-2								🟨	🟨	🟨	🟨	🟨	
		Display warning on GUI-MAP	IMP-T005-3												🟨	
		Implement export result	IMP-T005-4												🟨	
	Team final review	Each member review	IMP-T006-1		🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨		
		Final review meeting	IMP-T006-2												🟨	
		Make change base on feedback	IMP-T006-3												🟨	
Testing	Unit Test	Create unit test	Test-T001-1			🟥	🟥	🟥								
		Regular review	Test-T001-2			🟥	🟥	🟥								
	Integration Test	Define test cases	Test-T002-1						🟥	🟥	🟥					
		Meeting with client	Test-T002-2							🟥	🟥					
		Perform test product	Test-T002-3					🟥	🟥	🟥	🟥					
		Keep track all change	Test-T002-4							🟥	🟥	🟥	🟥	🟥		
		Recording all fail tests	Test-T002-5		🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	
User Manual	Draft	Create a draft	UM-T001-1									🟩	🟩			
	Final Version	Revise the draft and make final version	UM-T001-2											🟩	🟩	
Testing Report	Draft	Create a draft	TR-T001-1									🟫	🟫			
	Final Version	Revise the draft and make final version	TR-T001-2											🟫	🟫	
Demo	Practice	Test and Practice for final demo	D-T001-1												🟥	🟥

Figure 5: Gantt Chart To Manage Progress